IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Holloway et al.**	§	
	§	Group Art Unit: **2185**
Serial No. **10/713,725**	§	
	§	Examiner: **Savla, Arpan P.**
Filed: **November 13, 2003**	§	
	§	
For: **Method and System for Dynamically Storing Frequently Used Instructions Using a Separate Cache**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

39698
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on September 12, 2006.

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 50-0563. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 50-0563. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 50-0563.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation, Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-3, 5-8, 10-13, and 15.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: 4, 9, and 14.
2. Claims withdrawn from consideration but not canceled: None.
3. Claims pending: 1-3, 5-8, 10-13, and 15.
4. Claims allowed: None.
5. Claims rejected: 1-3, 5-8, 10-13, and 15.
6. Claims objected to: None.

C. CLAIMS ON APPEAL

The claims on appeal are: 1-3, 5-8, 10-13, and 15.

STATUS OF AMENDMENTS

There are no amendments after the Final Office Action, which included the final rejections of the claims, that was mailed on June 16, 2006.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

Claim 1 is directed to a cache system for a computer system, comprising: a first cache for storing a first plurality of instructions; (Specification page 9, lines 16-26.) a second cache for storing a second plurality of instructions; (Specification page 9, lines 16-26.) wherein each instruction of the first plurality has an associated counter, and wherein when a first instruction of the first plurality is accessed, a first associated counter is incremented; (Specification page 9, lines 16-26.) and wherein when the first associated counter reaches a threshold, the first instruction of the first plurality is copied into the second cache. (Specification page 10, lines 5-11.)

B. CLAIM 2 - DEPENDENT

Claim 2 depends from claim 1 and is further directed to wherein each instruction of the second plurality has an associated counter, and wherein when an instruction of the second plurality is accessed, all other counters of the second plurality are decremented. (Specification page 13, lines 10-18.)

C. CLAIM 5 - DEPENDENT

Claim 5 depends from claim 1 and is further directed to wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy. (Specification page 10, lines 5-11.)

D. CLAIM 6 - INDEPENDENT

Claim 6 is directed to a method of managing cache in a computer system, comprising the steps of: checking for a first instruction in a first cache, wherein each instruction in the first cache has an associated counter; (Specification page 12, lines 7-13.) if the first instruction is found in

the first cache, incrementing a first associated counter; (Specification page 12, lines 7-13.) comparing a value of the first associated counter to a threshold; (Specification page 12, lines 7-13.) if the first associated counter exceeds the threshold, moving the first instruction from the first cache to a second cache. (Specification page 12, lines 7-13.)

E. CLAIM 8 - DEPENDENT

Claim 8 depends from claim 6 and is further directed to wherein each instruction of the second cache has an associated counter, and wherein when an instruction of the second cache is accessed, all other counters of the second cache are decremented. (Specification page 13, lines 10-18.)

F. CLAIM 10 - DEPENDENT

Claim 10 depends from claim 6 and is further directed to wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy. (Specification page 10, lines 5-11.)

G. CLAIM 11 - INDEPENDENT

Claim 11 is directed to a computer program product in a computer readable medium, comprising: (Specification page 16, lines 10-28.) first instructions for checking for a first line of data in a first cache, wherein each line of data in the first cache has an associated counter; (Specification page 12, lines 7-13.) second instructions for, if the first line of data is found in the first cache, incrementing a first associated counter; (Specification page 12, lines 7-13.) third instructions for comparing a value of the first associated counter to a threshold; (Specification page 12, lines 7-13.) fourth instructions for, if the first associated counter exceeds the threshold, moving the first line of data from the first cache to a second cache. (Specification page 12, lines 7-13.)

H. CLAIM 13- DEPENDENT

Claim 13 depends from claim 11 and is further directed to wherein each line of data of the second cache has an associated counter, and wherein when a line of data of the second cache is accessed, all other counters of the second cache are decremented. (Specification page 13, lines 10-18.)

I. CLAIM 15 - DEPENDENT

Claim 15 depends from claim 11 and is further directed to wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy. (Specification page 10, lines 5-11.)

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

A. GROUND OF REJECTION 1 (Claims 1, 3, and 6-7)

Whether claims 1, 3, and 6-7 are anticipated under 35 U.S.C. § 102(b) by U. S. Patent Application Publication 2002/0095553, published by *Mendelson*;

B. GROUND OF REJECTION 2 (Claims 2 and 8)

Whether claims 2 and 8 are obvious under 35 U.S.C. § 103(a) over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of U. S. Patent Application Publication 2001/0001873, published by *Wickraad*;

C. GROUND OF REJECTION 3 (Claims 5 and 10)

Whether claims 5 and 10 are obvious under 35 U.S.C. § 103(a) over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers", published by *Jouppi*;

D. GROUND OF REJECTION 4 (Claims 11-12)

Whether claims 11-12 are obvious under 35 U.S.C. § 103(a) over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Structured Computer Organization, 2nd Edition", published by *Tanenbaum*;

E. GROUND OF REJECTION 5 (Claim 13)

Whether claim 13 is obvious under 35 U.S.C. § 103(a) over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Structured Computer Organization, 2nd Edition", published by *Tanenbaum*, and further in view of U. S. Patent Application Publication 2001/0001873, published by *Wickraad*; and

F. GROUND OF REJECTION 6 (Claim 15)

Whether claim 15 is obvious under 35 U.S.C. § 103(a) over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Structured Computer Organization, 2nd Edition", published by *Tanenbaum*, and further in view of "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers", published by *Jouppi*.

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1, 3, and 6-7)

The Examiner rejected claims 1, 3, and 6-7 under 35 U.S.C. § 102(b) as being anticipated by U. S. Patent Application Publication 2002/0095553, published by *Mendelson*. This position is not well-founded.

A.1. Claim 1

Applicants' claim 1 recites a first cache for storing a first plurality of instructions and a second cache for storing a second plurality of instructions. Each instruction of the first plurality has an associated counter. When a first instruction of the first plurality is accessed, its first associated counter is incremented.

When the first associated counter reaches a threshold, the first instruction is copied into the second cache. The first instruction is copied when the first associated counter reaches a threshold. As is clear from the claim language, the copying is not discretionary. The copying of the first instruction is required when the first associated counter reaches the threshold.

The claim also clearly defines when that required copying must occur. The first instruction is copied when the first associated counter reaches the threshold. According to Applicants' claim 1, the copying occurs at the time the counter merely reaches the threshold. Thus, copying occurs prior to the counter exceeding the threshold.

Mendelson teaches a filter trace cache (FTC) and a main trace cache (MTC). Newly received traces are placed in the FTC. At a later time, a trace may be, but is not required to be, evicted from the FTC. If a trace is evicted, the evicted trace will either be discarded or moved to the MTC. The decision as to whether to discard a trace or move it to the MTC will depend on that trace's access counter value. If the value is below a threshold, the trace is discarded. If the value is equal to or above a threshold, the trace is moved to the MTC.

Traces can stay in the FTC indefinitely, regardless of their counter value. Traces stay as long as new traces do not evict them. See *Mendelson*, page 3, paragraph 0033. Thus, a trace stays in the FTC until a new trace is received that must replace the existing trace. If, for example, a trace is in the FTC and no new traces are received, that existing trace will stay in the

FTC indefinitely, regardless of the existing trace's counter value.

A trace will stay in the FTC regardless of its counter value. Thus, the trace in the example above, where no new traces are received, will stay in the FTC while its counter value continues to increase well beyond the threshold value. A trace is only evicted from the FTC when a new trace needs to be stored in the existing trace's location. Only when a new trace is received that needs to replace an existing trace is a determination made as to whether to discard the existing trace or to move the existing trace to the MTC.

Traces are not moved out of the FTC because their values reached a threshold. No action is taken to move a trace out of the FTC based on the trace's counter value. A trace is only moved out of the FTC when that trace is evicted by a new trace. The counter value plays no part in the decision as to whether or not a trace will be evicted. The counter value is used to determine what to do with a trace once it has been evicted.

A decision as to whether to discard the existing trace or move the existing trace requires evaluating the counter. This decision is only made once an existing trace is evicted. This decision is not made prior to the trace being evicted. Prior to the trace being evicted, the trace remains in the FTC regardless of the counter value. The counter can continue to increase indefinitely, well beyond the threshold.

Applicants' claim 1 describes "when the first associated counter reaches a threshold, the first instruction of the first plurality is copied into the second cache". Thus, according to Applicants' claim, the first instruction is copied when the counter reaches a threshold. No other action is involved. When the counter reaches the threshold, the first instruction is copied. Unlike *Mendelson* where traces can stay in the FTC after their counter value reaches a threshold, according to Applicants' claims, the first instruction cannot stay in the first cache after its counter reaches a threshold. The first instruction must be copied once its counter reaches the threshold.

In responding to Applicants' argument, the Examiner states that it is inherent that a comparison is made between the first associated counter and the threshold claimed by Applicants' in claim 1. The Examiner goes on to state that Applicants' claim 1 does not specify when such a comparison is made. See Final Office Action, mailed June 16, 2006, page 13.

Because the Examiner does not find an explicit reference in Applicants' claim 1 to a time to make a comparison, the Examiner disregards the claim language "when the first associated counter reaches". The Examiner appears to be interpreting Applicants' claim 1 as claiming

copying an instruction, at some specific defined comparison time, if the instruction's counter exceeds a threshold. This is not what is recited by Applicants' claim 1.

Applicants' do not claim a specific time to make a comparison. Applicants claim copying when the counter reaches a threshold. The broadest most reasonable interpretation of Applicants' claim 1 is that a comparison is continuous so that the instance the counter reaches a threshold, the first instruction is copied. *Mendelson* does not teach copying an instruction the instance the instruction's counter reaches a threshold.

Mendelson does not teach a trace being copied when its value reaches a threshold. In *Mendelson*, a trace can remain in the FTC with its value well above the threshold. Therefore, *Mendelson* does not teach when the first associated counter reaches a threshold, the first instruction of the first plurality is copied into the second cache. Therefore, *Mendelson* does not anticipate Applicants' claim 1.

A.2. Claim 6

Applicants' claim 6 recites a method of managing cache in a computer system, comprising the steps of: checking for a first instruction in a first cache, wherein each instruction in the first cache has an associated counter; if the first instruction is found in the first cache, incrementing a first associated counter; comparing a value of the first associated counter to a threshold; and if the first associated counter exceeds the threshold, moving the first instruction from the first cache to a second cache.

Mendelson teaches each trace having an access counter that counts accesses to the trace since it was inserted into a particular trace cache. See *Mendelson*, paragraph 0030. In contradistinction, Applicants' claim 6 recites incrementing a counter if an instruction is found in the cache. Finding a trace in a cache and accessing a trace in a cache are not the same.

A trace could be found in a cache but never accessed. As a result, counting only accesses would result in a lower counter value than counting each time a trace was found in the cache.

Mendelson requires a trace to be accessed before the trace's counter is incremented. According to Applicants' claim 6, a counter would be incremented even when an instruction is found in the cache, but never accessed.

Mendelson does not teach checking for an instruction in a cache, and then incrementing a counter if the instruction is found in the cache. *Mendelson* teaches counting the number of times an

actual access of a trace occurs. Merely finding a trace in the cache does not result in incrementing the counter according to *Mendelson*. The trace must be accessed.

Because *Mendelson* does not teach checking for an instruction in a cache, and then incrementing a counter if the instruction is found in the cache, *Mendelson* does not anticipate Applicants' claim 6.

B. GROUND OF REJECTION 2 (Claims 2 and 8)

The Examiner rejected claims 2 and 8 under 35 U.S.C. § 103(a) as being obvious over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of U. S. Patent Application Publication 2001/0001873, published by *Wickraad*. This position is not well-founded.

B.1. Claim 2

Applicants' claim 2 recites the cache system of claim 1, wherein each instruction of the second plurality has an associated counter, and wherein when an instruction of the second plurality is accessed, all other counters of the second plurality are decremented. The second plurality of instructions are stored in the second cache.

The Examiner refers to *Wickraad*, paragraph 0013, and states that "it should also be noted that if the 'memory operand' is loaded into the cache line it is inherently required the 'memory operand' was first accessed from main memory." See Final Office Action mailed June 16, 2006. *Wickraad*, paragraph 0013, teaches a least recently used (LRU) algorithm. When a memory operand is to be loaded into a cache, it is loaded into the cache line that is the least recently used. The memory operand cache line is then assigned a counter value indicating that it is the most recently used and the counter values of the existing lines are decremented.

According to Applicants' claim 2, a second plurality of instructions is stored in a second cache. When one of the second instructions is accessed, the counters of all of the other second instructions are decremented. It is important to note that the second instruction claimed by Applicants is already stored in the second cache when it is accessed. When this second instruction is accessed, the counters of all of the other second instructions are then decremented.

In contradistinction, the memory operand taught by *Wickraad* is not stored in the cache when it is accessed. It is accessed, and then, it is stored in the cache. After it is stored in the cache,

the counters of the other cache lines are decremented. *Wickraad* teaches merely where this memory operand will be stored in the cache after it is accessed.

Because the combination of *Mendelson* and *Wickraad* does not teach an instruction that is already stored in the cache being accessed, resulting in decrementing the counters of the other instructions that are stored in the cache, the combination does not render Applicants' claim 2 obvious.

B.2. Claim 8

Applicants' claim 8 recites the method of claim 6, wherein each instruction of the second cache has an associated counter, and wherein when an instruction of the second cache is accessed, all other counters of the second cache are decremented.

Mendelson does not teach checking for an instruction in a cache, and then incrementing a counter if the instruction is found in the cache. *Mendelson* teaches counting the number of times an actual access of a trace occurs. Merely finding a trace in the cache does not result in incrementing the counter according to *Mendelson*. The trace must be accessed.

The combination of *Mendelson* and *Wickraad* does not render Applicants' claim 8 obvious because the combination does not teach or suggest checking for an instruction in a cache, and then incrementing a counter if the instruction is found in the cache in combination with each instruction of the second cache having an associated counter, and wherein when an instruction of the second cache is accessed, all other counters of the second cache are decremented.

C. GROUND OF REJECTION 3 (Claims 5 and 10)

The Examiner rejected claims 5 and 10 under 35 U.S.C. § 103(a) as being obvious over U.S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers", published by *Jouppi*. This position is not well-founded.

C.1. Claim 5

Applicants' claim 5 recites the cache system of claim 1, wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used

policy.

The Examiner states that *Mendelson* does not teach the second cache being fully associative and following a least recently used policy. The Examiner relies on *Jouppi* to teach this feature.

The combination of *Mendelson* and *Jouppi* does not render Applicants' claim 5 obvious because the combination does not teach the first instruction being copied when the first associated counter reaches a threshold and the first cache being an instruction cache and the second cache being fully associative and following a least recently used policy.

C.2. Claim 10

Applicants' claim 10 recites the method of claim 6, wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy.

The Examiner states that *Mendelson* does not teach the second cache being fully associative and following a least recently used policy. The Examiner relies on *Jouppi* to teach this feature.

Mendelson does not teach checking for an instruction in a cache, and then incrementing a counter if the instruction is found in the cache. *Mendelson* teaches counting the number of times an actual access of a trace occurs. Merely finding a trace in the cache does not result in incrementing the counter according to *Mendelson*. The trace must be accessed.

The combination of *Mendelson* and *Wickraad* does not render Applicants' claim 8 obvious because the combination does not teach or suggest checking for an instruction in a cache, and then incrementing a counter if the instruction is found in the cache in combination with wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy.

D. GROUND OF REJECTION 4 (Claims 11-12)

The Examiner rejected claims 11-12 under 35 U.S.C. § 103(a) as being obvious over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Structured Computer Organization, 2nd Edition", published by *Tanenbaum*. This position is not well-founded.

The Examiner relies on *Tanenbaum* to teach that hardware and software are logically equivalent. The Examiner relies on *Mendelson* to teach the remaining features of Applicants'

claims 11 and 12.

Applicants' claim 11 recites a computer program product in a computer readable medium, comprising: first instructions for checking for a first line of data in a first cache, wherein each line of data in the first cache has an associated counter; second instructions for, if the first line of data is found in the first cache, incrementing a first associated counter; third instructions for comparing a value of the first associated counter to a threshold; fourth instructions for, if the first associated counter exceeds the threshold, moving the first line of data from the first cache to a second cache.

Mendelson teaches each trace having an access counter that counts accesses to the trace since it was inserted into a particular trace cache. See *Mendelson*, paragraph 0030. In contradistinction, Applicants' claim 11 recites if the first line of data is found in the first cache, incrementing a first associated counter. Finding a trace in a cache and accessing a trace in a cache are not the same.

A trace could be found in a cache but never accessed. As a result, counting only accesses would result in a lower counter value than counting each time a trace was found.

Mendelson requires a trace to be accessed before the trace's counter is incremented. According to Applicants' claim 11, a counter would be incremented even when a first line of data is found in the cache, but never accessed.

Mendelson does not teach checking for a first line of data in a first cache, wherein each line of data in the first cache has an associated counter; second instructions for, if the first line of data is found in the first cache, incrementing a first associated counter. *Mendelson* teaches counting the number of times an actual access of a trace occurs. Merely finding a trace in the cache does not result in incrementing the counter according to *Mendelson*. The trace must be accessed.

Because *Mendelson* does not teach checking for a first line of data in a first cache, wherein each line of data in the first cache has an associated counter; second instructions for, if the first line of data is found in the first cache, incrementing a first associated counter, the combination of *Mendelson* and *Tanenbaum* does not render Applicants' claims 11 and 12 obvious.

E. GROUND OF REJECTION 5 (Claim 13)

The Examiner rejected claim 13 under 35 U.S.C. § 103(a) as being obvious over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Structured Computer Organization, 2nd Edition", published by *Tanenbaum*, and further in view of U. S. Patent Application Publication 2001/0001873, published by *Wickraad*. This position is not well-founded.

Applicants' claim 13 depends from claim 11 and further recites wherein each line of data of the second cache has an associated counter, and wherein when a line of data of the second cache is accessed, all other counters of the second cache are decremented.

The Examiner states that the combination of *Mendelson* and *Tanenbaum* teaches the limitations of claim 13 but does not teach wherein the instruction of the second cache has an associated counter, and wherein when in instruction of the second cache is accessed, all other counters of the second cache are decremented.

The combination of *Mendelson*, *Tanenbaum*, and *Wickeraad* does not render Applicants' claim 13 obvious because the combination does not teach checking for a first line of data in a first cache, wherein each line of data in the first cache has an associated counter; second instructions for, if the first line of data is found in the first cache, incrementing a first associated counter, in combination with wherein each line of data of the second cache has an associated counter, and wherein when a line of data of the second cache is accessed, all other counters of the second cache are decremented.

F. GROUND OF REJECTION 6 (Claim 15)

The Examiner rejected claim 15 under 35 U.S.C. § 103(a) as beings obvious over U. S. Patent Application Publication 2002/0095553, published by *Mendelson* in view of "Structured Computer Organization, 2nd Edition", published by *Tanenbaum*, and further in view of "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers", published by *Jouppi*. This position is not well-founded.

Applicants' claim 15 recites the computer program product of claim 11, wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy.

The Examiner states that the combination of *Mendelson* and *Tanenbaum* teaches the features of claim 15 but does not teach the second cache being fully associative and following a least recently used policy. The Examiner relies on *Jouppi* to teach this feature.

The combination of *Mendelson*, *Tanenbaum*, and *Jouppi* does not render Applicants' claim 15 obvious because the combination does not teach checking for a first line of data in a first cache, wherein each line of data in the first cache has an associated counter; second instructions for, if the first line of data is found in the first cache, incrementing a first associated counter, in combination with wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy.

G. CONCLUSION

The rejections of Applicants' claims are in error and should be reversed for the reasons given above.

/Lisa L.B. Yociss/

Lisa L.B. Yociss
Reg. No. 36,975
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A cache system for a computer system, comprising:

a first cache for storing a first plurality of instructions;

a second cache for storing a second plurality of instructions;

wherein each instruction of the first plurality has an associated counter, and wherein when a first instruction of the first plurality is accessed, a first associated counter is incremented; and

wherein when the first associated counter reaches a threshold, the first instruction of the first plurality is copied into the second cache.

2. The cache system of claim 1, wherein each instruction of the second plurality has an associated counter, and wherein when an instruction of the second plurality is accessed, all other counters of the second plurality are decremented.

3. The cache system of claim 1, wherein the first instruction of the first plurality is accessed from the second cache.

5. The cache system of claim 1, wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy.

6. A method of managing cache in a computer system, comprising the steps of:

checking for a first instruction in a first cache, wherein each instruction in the first cache has an associated counter;

if the first instruction is found in the first cache, incrementing a first associated counter;

comparing a value of the first associated counter to a threshold;

if the first associated counter exceeds the threshold, moving the first instruction from the first cache to a second cache.

7. The method of claim 6, further comprising the step of:

accessing the first instruction from the second cache.

8. The method of claim 6, wherein each instruction of the second cache has an associated counter, and wherein when an instruction of the second cache is accessed, all other counters of the second cache are decremented.

10. The method of claim 6, wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy.

11. A computer program product in a computer readable medium, comprising:

first instructions for checking for a first line of data in a first cache, wherein each line of data in the first cache has an associated counter;

second instructions for, if the first line of data is found in the first cache, incrementing a first associated counter;

third instructions for comparing a value of the first associated counter to a threshold;

fourth instructions for, if the first associated counter exceeds the threshold, moving the first line of data from the first cache to a second cache.

12. The computer program product of claim 11, further comprising the step of:

accessing the first line of data from the second cache.

13. The computer program product of claim 11, wherein each line of data of the second cache has an associated counter, and wherein when a line of data of the second cache is accessed, all other counters of the second cache are decremented.

15. The computer program product of claim 11, wherein the first cache is an instruction cache and the second cache is fully associative and follows a least recently used policy.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.